Fly-The-Bee: A game imitating concept learning in bees

D. Kleyko *, E. Osipov, M. Björk, H. Toresson, and A. Öberg

Luleå University of Technology, 971 87 Luleå, Sweden
denkle@ltu.se; eao@ltu.se; magbjr@ltu.se; hentho@ltu.se; antobe@ltu.se;

Abstract
This article presents a web-based game functionally imitating a part of the cognitive behavior of a living organism. This game is a prototype implementation of an artificial online cognitive architecture based on the usage of distributed data representations and Vector Symbolic Architectures. The game demonstrates the feasibility of creating a lightweight cognitive architecture, which is capable of performing rather complex cognitive tasks. The cognitive functionality is implemented in about 100 lines of code and requires few tens of kilobytes of memory for its operation, which make the concept suitable for implementing in low-end devices such as minirobots and wireless sensors.

Keywords: Vector Symbolic Architecture, distributed data representation, concept learning, cognition

1 Introduction

This article presents a prototype of an artificial online learning system, which imitates the concept learning in a living organism. The approach is inspired by the experiments with honey bees, which demonstrate their capabilities of learning new concepts [1], [2]. In short the experiment is about training bees to select one or another direction in a maze illustrated in Figure 1 based on the visual input (a certain pattern of two geometrical figures with specific attributes arranged in a certain order on a plane) indicating the direction towards a reward (sucrose), the other direction resulted in a penalty (quinine). The experiments were designed to use reinforcement learning in the training phase. The challenges associated with designing an artificial system, which would imitate bees' concept learning include: a.) An ability of the system to be trained on small numbers of training trials (the bees mastered concepts after 30 trials only); b.) An ability for continuous learning; and c.) Flexibility in interpretation of the previously unseen visual targets [1]. The presented prototype is based on the design of a generic artificial learning pipeline described in [3], which is anchored to psychological models of similarity [4] and features a concept encoding and reasoning using distributed data representation and Vector Symbolic Architectures (VSA) [5].

The prototype is built in the form of an online web-based game. The scenario of the game essentially replicates the real-life experiments with honey bees. The objective of the game is

*Corresponding author

to attract a bee-avatar launched from the server to one of two players by demonstrating it visual patterns and treating it with a reward. The Fly-the-Bee game is accessible for trial by the following address `http://FlyTheBee.ddns.net`. The front-end for the **moderator** is accessible via `http://FlyTheBee.ddns.net/admin`. Importantly, the entire learning algorithm requires only few tens of kilobytes of memory for operation, which makes the concept suitable for implementation in devices with limited computational resource, for example miniature robots or embedded devices.

The article proceeds as follows. Section 2 presents an overview of the related work. The relevant aspects of VSA are introduced in section 3. Section 4 describes games scenario and the outline of the VSA concept learning pipeline behind its implementation. Section 5 presents details of implementation and simulations. The paper is concluded in Section 6.

2 Related Work

This article does not concern with benchmarking of the presented in this article prototype of an artificial learning system with other self-learning, AI-enabled games or cognitive architectures in general. The major objective is to present a proof-of-concept implementation of a VSA-enabled online reasoning platform and demonstrate its functional match to a part of the cognitive behavior of a living organism. As such, the overview of the related work resorts to the discussion of concepts specific to the concepts behind the presented implementation.

The first results providing the evidence that honey bees can learn relations related visual stimuli were documented in [2]. This work showed that honey bees are able to learn the *sameness-difference* concepts for visual stimuli via delayed sample (non) matching to initial sample. Later in [6] it was shown that bees are capable of learning the *above-below* relation in visual stimuli and able to apply learned relation in transfer tests. [1] presents the results of experiments showing that honey bees can master two concepts simultaneously, namely the *above-below* or *left-right* and *sameness-difference*. An overview of results related to conceptual learning by miniature brains is presented in [2]. A potential usage of the findings for computer vision is discussed in [7]. A discussion of implications of the results in the context of neuromorphic systems is presented in [8].

Distributed data representation is widely used in computer based semantic reasoning [5], [9]. Examples of capabilities of distributed representations for solving cognitive tasks include solving Raven's progressive matrices [10], [11]. They could also be applied for representation of heterogeneous sensory stimulus [12]. In this article distributed representations and in particular Vector Symbolic Architectures [13] are applied for online concept learning and reasoning.

3 Relevant aspects of Vector Symbolic Architectures

Vector Symbolic Architecture is an approach for encoding and operations on distributed representation of information, and previously has mainly been used in the area of cognitive computing for representing structured knowledge and reasoning upon it [5, 13]. Using distributed representations all entities such as concept labels, objects, attributes are represented by random codewords of very high dimension, i.e. several thousand bits. Further in this article such vectors are also referred to as HD-codes. Putting an object into a correspondence with a certain concept is done via bit-wise XOR operation on HD-codes representing the concept's label (referred to as *role* in the VSA terminology) and the object (called *filler* in the VSA terminology). For example a statement "Object X is a star" is represented as $\mathbf{STAR}_{HD} \oplus \mathbf{X}_{HD}$. The formation of

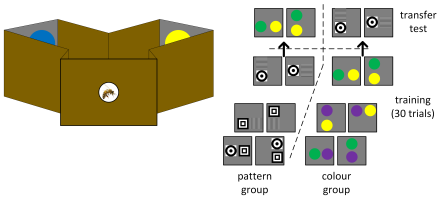

Figure 1: The maze and patterns used in experiments with honey bees.

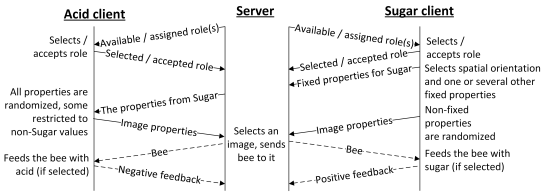

Figure 2: A graphical representation of the game's protocol.

a predicate consisting of several statements, for example "Object *star* is above object *circle*", is implemented by a thresholded sum of the HD-codes representing the elements of the predicate. Namely, the predicate describing a relation in the "star-above-circle" example would be represented as $REL_{above-below} = [\mathbf{LABEL}_{above-below} + \mathbf{STAR}_{HD} \oplus \mathbf{X}_{HD} + \mathbf{CIRCLE}_{HD} \oplus \mathbf{Y}_{HD}]$. The notation $[A + B + C]$ is used to indicate a bit-wise thresholded sum (also called a *majority* sum) of n vectors results in 0 when $n/2$ or more arguments are 0, and 1 otherwise. Each predicate in the form above is a symbolic vector. There is a defined scalar metric of similarity between two binary vectors, namely *Hamming distance*. A smaller Hamming distance between two vectors indicates larger content/structural similarity between them.

The high information capacity of the distributed data representation (i.e. a capability of joining many predicates into a single representation) and the existence of the well-defined similarity metric make VSA suitable for implementation of associative memory with self-learning capabilities. The VSA-based associative memory is filled using the learning by example. That is during the system operation predicates of conditions leading to the particular reward are VSA encoded and added a single HD code which characterizes the system's experience. This operation is called *mapping*. Due to the statistical properties of the distributed representations, the dominating concepts associated with the specific system's outcome become dominating in this joint representation. Next time a system experiences stimuli (e.g. a visual scene), which representation is similar (to a certain threshold value) to the one in the associative memory, it may make a prediction about the outcome. The next section presents an example of a concept formation and the mapping operation.

4 Scenario of the game and the learning pipeline

The scenario of the game is graphically illustrated in Figure 2. In the game there are two human players, aka "Sugar" and "Acid", one robot-server and a web front-end for the moderator. In short the game is about attracting a virtual bee launched by the robot server to the player, who gives a positive reward, i.e. sugar. The bee is attracted by demonstrating visual patterns, which are not known to the robot server when the game starts. The patterns shown by the two players are different between each other (this is assured by blacklisting some relations chosen by the "Sugar" player at the "Acid" side). The patterns for the particular player bear common features, which the system must learn. Other features are randomized, meaning that the visual scenes presented to the virtual bee are never identical.

The game goes on in rounds and on each round the bee is presented with a new scene by the players. The virtual bee flies to one or the other player. The player treats the Bee with the corresponding nutrient (either sugar or acid). Initially, the virtual bee chooses its destination

at random. However, as the game continues the bee collects an experience by mapping the visual pattern to the type of the treatment. After only a few trials the bee begins to prefer the "Sugar" player. The bee is continuously learning during the game. The learned experience is transferable, meaning that when a moderator changes roles (i.e. the "Acid" player starts to generate patterns of "Sugar"), the robot bee will choose patterns from the "Acid" player automatically.

The algorithm of the virtual bee implements a simple, yet extensible online learning pipeline. Due to the space limitations only an overview of its major functions is given here, for more details an interested reader is referred to [3]. The pipeline begins with a *vision circuitry*, which does the features' extraction from of the visual input. The scenes in the Fly-the-Bee game are similar to the images in the real-world experiment, i.e. each scene contains geometrical figures with single-color fill as exemplified in Figure 1.While in general the *vision circuitry* block could be implemented using the standard image processing toolboxes, in the game the set of features is deterministic, therefore *no* actual image processing happens at player's devices.

The next and the major part of the pipeline is the *scene encoding block*, which encodes the relations between the objects and their features in the observed scene. In the game each scene was encoded as a combination of two types of relations: *explicit* and *implicit*. The explicit relation signifies the objects engaged in this relation, for example "Object A is above object B". The implicit relation indicates only the fact that the relation exists, without signifying how the objects are engaged in this relation. When, for example, the *vision circuitry* detects that object A is larger than object B the scene encoding block creates an implicit relation "is larger" without specifying which of the objects object is larger. This taxonomy of relations is inspired by the psychological model of similarity [4] and captures different aspects of the similarity assessment: *contrast assessment*, which cross-compares the sets of scenes' features and *structural mapping*, which analyses the relational commonalities and differences between the scenes' objects. In the Fly-the-Bee game the virtual bee is capable of learning the following set of left-right, above-below, larger-smaller, same-different. This is achieved by the following VSA-encoding of the scene (note, that the form of encoding above is the particular instance of the generic encoding template presented in [3]):

$\mathbf{SCENE}_i = [\mathbf{spatial} + \mathbf{orient}_1 \oplus \mathbf{Obj}_X + \mathbf{orient}_2 \oplus \mathbf{Obj}_Y + \mathbf{ls} \oplus \mathbf{SizeRel}_i + \mathbf{sd} \oplus \mathbf{SameRel}_i]$.

The first part of the representation $\mathbf{spatial} + \mathbf{orient}_1 \oplus \mathbf{Obj}_X + \mathbf{orient}_2 \oplus \mathbf{Obj}_Y$ describes the spatial relation of the two object where **spatial** will be assigned an HD-code for either *above-below* or *left-right* relation as they appear in the game; $\mathbf{orient}_i$ are HD-roles for the specific placement, e.g. "is above"; finally, $\mathbf{Obj}_i$ is an HD-code (filler) for the objects in the corresponding placement. The parts $\mathbf{ls} \oplus \mathbf{SizeRel}_i$ and $\mathbf{sd} \oplus \mathbf{SameRel}_i$ encode implicit relations ("is larger-smaller" and "is same-different").

When scenes are VSA-encoded as shown above they are temporarily stored awaiting the reward signal to arrive in order to complete the process of experience forming. When a reward arrives the VSA representation of the present experience is updated as $\mathbf{EXPERIENCE} = [\mathbf{EXPERIENCE} + \mathbf{SCENE}_i \oplus \mathbf{REWARD}]$. While the system could obviously store VSA representations of experiences for both the positive and the negative rewards, in the Fly-the-Bee game only the *positive* experience is kept in the associative memory.

The system implements reasoning via the *recall* of **EXPERIENCE** stored in the associative memory. In the recall process the VSA-encoded scenes of two players ($\mathbf{SCENE}_1$ and $\mathbf{SCENE}_2$) are presented to the system. The virtual bee retrieves the HD-code for the reward by *unbinding* it from the accumulated VSA-experience as:

$\mathbf{REWD}_1^* = \mathbf{EXPERIENCE} \oplus \mathbf{SCENE}_1$. $\mathbf{REWD}_2^* = \mathbf{EXPERIENCE} \oplus \mathbf{SCENE}_2$.

Finally, the virtual bee decides on the direction where to fly selecting the player who showed

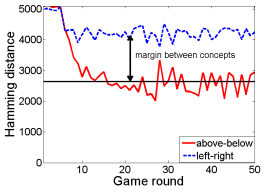

Figure 3: Test 1.

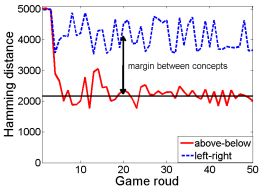

Figure 4: Test 2.

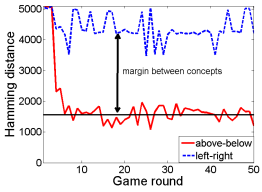

Figure 5: Test 3.

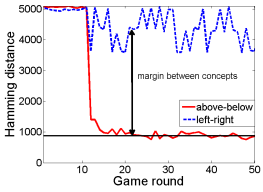

Figure 6: Test 4.

the scene whose Hamming distance is smaller to the HD-code representing the positive reward.

5 Details of implementation, run-time performance

The Fly-the-Bee game is implemented using Node.js platform, HTML5, JavaScript. The visual scene for each player is characterized by seven attributes: the spatial placement of objects, the shapes, the colors and the sizes. The placement attribute has two alternatives, i.e. left-right or above-below placement; each of the shapes, colors and sizes attributes had five alternative values - 15 in total. The players have a possibility to choose the number of attributes which will be fixed in the scenes, the not fixed attributes are randomized for each new game round. The "Sugar" player configured its rules for the scene generation first. The chosen configuration is blacklisted for the "Acid" user for the training consistency reasons.

The virtual bee implementation used 31 randomly generated HD-codes (10000 binary elements each): 15 for representing different values of object's attributes; 3 for representing roles in the VSA encoding; 12 for representing the labels *above-below*, *left-right*, *larger-smaller* and *same-different* relations; and 1 for representing the reward signal. The moderator's GUI graphically displays the log of game activities and is capable of forcing the role change for testing the transferability of the experience.

The JavaScript implementation of the concept learning by the virtual bee is about 100 lines of code. The associative memory was implemented as a list of 10000 element byte-arrays. In total 44 elements were created for implementing the learning and reasoning. The entire process of forming the representation of scenes and the recall-based reasoning takes about 50 operations with HD-codes (XOR and majority sum). Note that the byte-array is the smallest possible array implementation in Javascript. Obviously if operations would be implemented on bits-granularity the memory footprint could be reduced drastically, i.e. the entire associative memory of the bit-level virtual bee could fit in 44 kB of memory.

5.1 Learning and reasoning performance

In order to assess the performance of the virtual bee learning curve and the reasoning accuracy the game was played with four different configurations of the concepts chosen by the "Sugar" player. In the first test scenario only spatial relation (*above-below*) was fixed and other attributes were chosen randomly. In the second test scenario two relations were fixed (*above-below* and *larger-smaller*). In the third scenario the set of fixed relations was extended by the *same-different* relation. Finally, in the forth test even shape attributes were fixed. Recall that at the same time the "Acid" player could generate any pattern of attributed not being fixed by the "Sugar" player. For each configuration 50 rounds of the game was played. The learning curves and the recall accuracy for all tests are shown in Figures 3 – 6.

The obtained results are positive. Initially during several few runs the virtual bee chooses the direction of fly at random as the system's experience transfers from a totally blank to the saturated. The more positive rewards are collected by the bee the more persistent is its choice of the correct direction. Notably and as expected the similarity threshold of the recall result to the clean HD-code for the positive reward is larger (i.e. Hamming distance is closer to zero) for cases where more attributes are fixed in the pattern.

6 Conclusion

This article presented a working prototype of an online artificial concept learning system, which functionally imitates a part of the cognitive behavior of a living organism - the concept learning by honey bees. The system adopts Vector Symbolic Architectures using high-dimensional binary codes for representing a scene, the associative memory and the reasoning. The prototype features the simplicity of the implementation and small memory footprint, which makes the adopted pipeline suitable for implementation even on units with severely constrained computing resources, i.e. miniature robots and embedded devices.

Acknowledgements. This work is supported by the Swedish Foundation for International Cooperation in Research and Higher Education (STINT), institutional grant IG2011-2025.

References

[1] Avarguès-Weber A., Dyer A. G., Combe M., and Giurfa M. Conceptual learning by miniature brains. *Proceedings of the National Academy of Sciences*, 109:7481–7486, 2012.

[2] Avarguès-Weber A. and Giurfa M. Simultaneous mastering of two abstract concepts with a miniature brain. *Proceedings of the Royal Society B: Biological Sciences*, 280:1–9, 2013.

[3] Kleyko D., Osipov E., Gayler R. W., Khan A. I., and Dyer A. G. Imitation of concept learning by honey bees using vector symbolic architectures. *Subm. to Bio. Ins. Cog. Architectures*, 2015.

[4] Markman A. and Gentner D. Nonintentional similarity processing. In *The New Unconscious.*, pages 107–137. Oxford University Press., 2005.

[5] Kanerva P. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159, 2009.

[6] Avarguès-Weber A., Dyer A. G., and Giurfa M. Conceptualization of above and below relationships by an insect. *Proceedings of the Royal Society B: Biological Sciences*, 278:898–905, 2011.

[7] et al. Dyer A. G. Flying in complex environments: Can insects bind multiple sensory perceptions and what could be the lessons for machine vision? *Jour. of Soft. Eng. and App.*, 7:406–412, 2014.

[8] et al. Sandin F. Concept learning in neuromorphic vision systems: What can we learn from insects? *Journal of Software Engineering and Applications*, 7:387–395, 2014.

[9] Plate T. A. *Holographic reduced representations: Distributed representation for cognitive structures*. Center for the Study of Language and Information (CSLI), 2003.

[10] Rasmussen D. and Eliasmith C. A neural model of rule generation in inductive reasoning. *Topics in Cognitive Science*, 3(1):140–153, 2011.

[11] et al. Levy S. D. Bracketing the beetle: How wittgenstein's understanding of language can guide our practice in agi and cognitive science. In *AGI, LNCS Vol. 8598*, pages 73–84, 2014.

[12] Kleyko D., Osipov E., Papakonstantinou N., Vyatkin V., and Mousavi A. Fault detection in the hyperspace: Towards intelligent automation systems. In *The proceedings of INDIN*, 2015.

[13] Gallant S. I. and Okaywe T. W. Representing objects, relations, and sequences. *Neural Computation*, 25(8):2038–2078, 2013.